WOLL2WOLL SOFTWARE

Release Notes – November 13th, 2023

FirePower X3

Supporting RAD Studio 12 Athens

© Woll2Woll Software Inc.

357 Yukon Way

Livermore, CA 94550

Email: sales@woll2woll.net

http://www.woll2woll.com

Table of Contents

RAD Studio 12 Notes 1

Skia Performance 1

Android Notes 1

WHATS NEW IN FIREPOWER X, X2, and X3 1

Skia Integration 1

Spell Checking Improvements 1

Edit improvements 2

TwwDataGrouper Component 2

TwwTreeView Component 5

Setting up the TwwTreeView to edit data 5

Configuring the type of control to edit with 6

Auto-sizing of Caption and Edit columns 6

Assign images to the caption column 7

Enabling multi-selection checkboxes 7

Common Methods 7

Adding Drag/Drop 7

Adding tree items with code. 7

Combo Enhancements 7

Customizing the dropdown icon 8

Enabling Multi-Selection Checkboxes 8

Select predefined listbox layouts for the dropdown list 8

TwwLayoutGrid modernization 9
TwwDataGrid now supports background images 9
See Revision History for changes in FirePower 9
Demos 10
FirePower's Masterpiece Grid 13
Versatile LayoutGrid/ListView 17
Transition Viewer 19
Flexible and Sophisticated TrackBars 20
Supports multiple thumbs 20
Custom Thumb bitmaps 20
Inverted display 20
Embeddable into FirePower Grids 20
Custom Styles 21
Display of Tick marks and labels 21
Lightning quick and powerful ListBoxes 22
Powerful and flexible Designers for Codeless Development 22
Stock Layouts 22
Incredibly fast loading and response times 22
Directly bind to data sources or your in-memory structures 22
Dynamically created data-entry forms 23
Superb Validation language 23
Regular Expression edit masks 24

Picture edit masks 24
Advanced LookupCombo and LookupDialog Controls 25
Quicken style incremental searching 25
Multiple Columns in the DropDown Box 25
Sorting flexibility 25
Embed into FirePower's DataGrid and LayoutGrid components: 26
Use unbound or bound 26
Date Control - TwwCalendarEdit 26
Advanced Combo Controls (TwwComboEdit, TwwAdvComboEdit) 26
TwwColorComboEdit 27
TwwButton 27
Advanced ImageControl with custom effects 27
Progress Activity Dialogs (Mobile) 28
Enhanced Sharing Services for Mobile 28
Checkboxes, Switches 28
Visual Filtering 29
Unmatched filtering power: 29
WildCard Filtering within Fields 29
Special customizable keywords 29
Filter memo fields 29

RAD Studio 12 Notes

Skia Performance

If you enable Skia for your projects and are deploying to mobile, we recommend setting GlobalUseSkia to false. Without this setting, we have noticed touch scrolling to be slower. See your project's .dproj file and you can change the setting there.

```
// Don't use skia for core functions, but support skia controls. This improves performance
  GlobalUseSkia := False;
  Application.Initialize;
```

Android Notes

If you have Android projects that were built with RAD Studio 11.2 or earlier, you may need to reset your libraries so that you don't get a compile error. The following NoSuchFileException error could occur otherwise regarding missing .jar files.

```
[PAClient Error] Error: E7688
```

```
E7688 java.nio.file.NoSuchFileException
```

`To reset your libraries, click on your project, and right click on your` *`Target Platforms | Android | Libaries`* `and select "Revert System Files to Default'`

WHATS NEW IN FIREPOWER X, X2, and X3

FirePower X3 brings a host of new capabilities to existing components, integration with new functionality in RAD Studio 12.

Skia Integration

FirePower has integrated use of Skia controls into its TwwLayoutGrid so you can embed them directly in them. It currently supports TskLabel and TskSvg

Spell Checking Improvements

The FirePower TwwMemoEdit now supports spell checking in mac, windows, and iOS environments using their native spell checkers. You can also add words to their respective dictionaries. When deploying to windows platforms, the editor seamlessly supports spell checking using the native windows dictionary in contrast to TMemo which does not support this.

For Android, you will need to provide your own dictionary (See MainDemo.SpellCheckDemo for an example) to integrate it with spelling. The TwwMemoEdit improves upon the spell checking in TMemo in that it supports multiple lines and improves upon the performance when having lengthy text.

End-User Notes: On desktop systems, you can right click the misspelled word to bring up corrections. On mobile devices you can press the misspelled word.

The native dictionary for mac is stored in your user's library\spelling\ folder

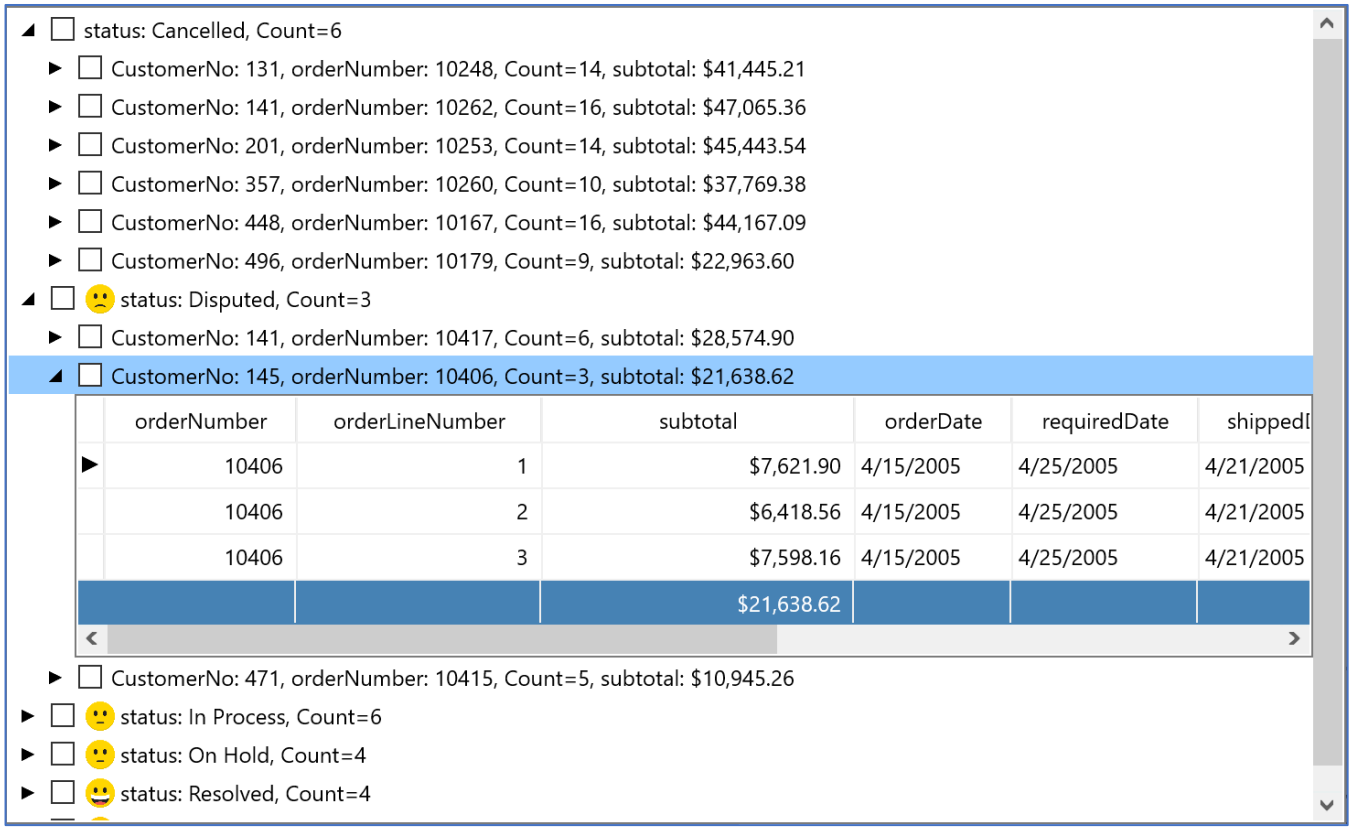
The native dictionary for windows is stored in your appdata\roaming\Microsoft\Spelling folder
On iOS, press on the misspelled words to get a menu of some options

On Android there is no native spell checker, but you can use the OnSpellCheck event to return if the word is valid. This demo uses a SQLite file Dictionary.db to check for the words.

Edit improvements

FirePower Edit controls updated to use the new TEdit style so that improvements to the FMX core components will manifest in FirePower controls

TwwDataGrouper Component



FirePower X2 introduced the TwwDataGrouper component which allows you to group your data according to field values in your dataset. You can use formulas to display summary data such as the count, sum, average, total, etc. The above is an example of the datagrouper display when grouping by the field status, and then CustomerNo;OrderNo. The way the DataGrouper works is to iterate through you're the records in your dataset and to create its display based on your assigned groupings and display parameters.

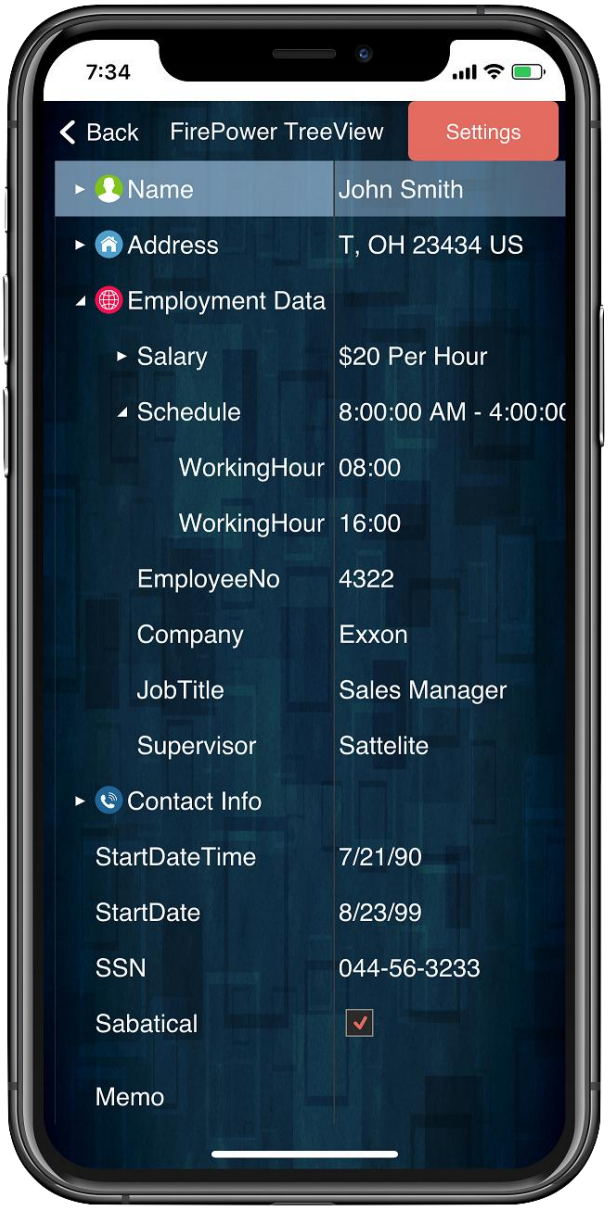
To being defining your groupings, use the *Groupings* property to define one or more TwwDataGroupingItem. You can use one or more fields for each group.

To control the display for each Grouping Item, see the TwwDataGroupItem.Display property that is part of each grouping item.

Here is another example with header columns.

OrderLineNo	Price Each	Quantity Ordered	Subtotal	orderDate
▸ status: Cancelled, subtotal: $238,854.18, Customers=6				
◢ status: Disputed, subtotal: $61,158.78, Customers=3				
◢ orderNumber: 10406				
◢ Danish Wholesale Imports, subtotal: $21,638.62, Number Orders=3				
▸ 1	$117.26	65	$7621.9	2005/04/15
▸ 2	$133.72	48	$6418.56	2005/04/15
▸ 3	$124.56	61	$7598.16	2005/04/15
▸ orderNumber: 10415				
▸ orderNumber: 10417				
▸ status: In Process, subtotal: $135,271.52, Customers=6				
▸ status: On Hold, subtotal: $169,575.61, Customers=4				

TwwTreeView Component



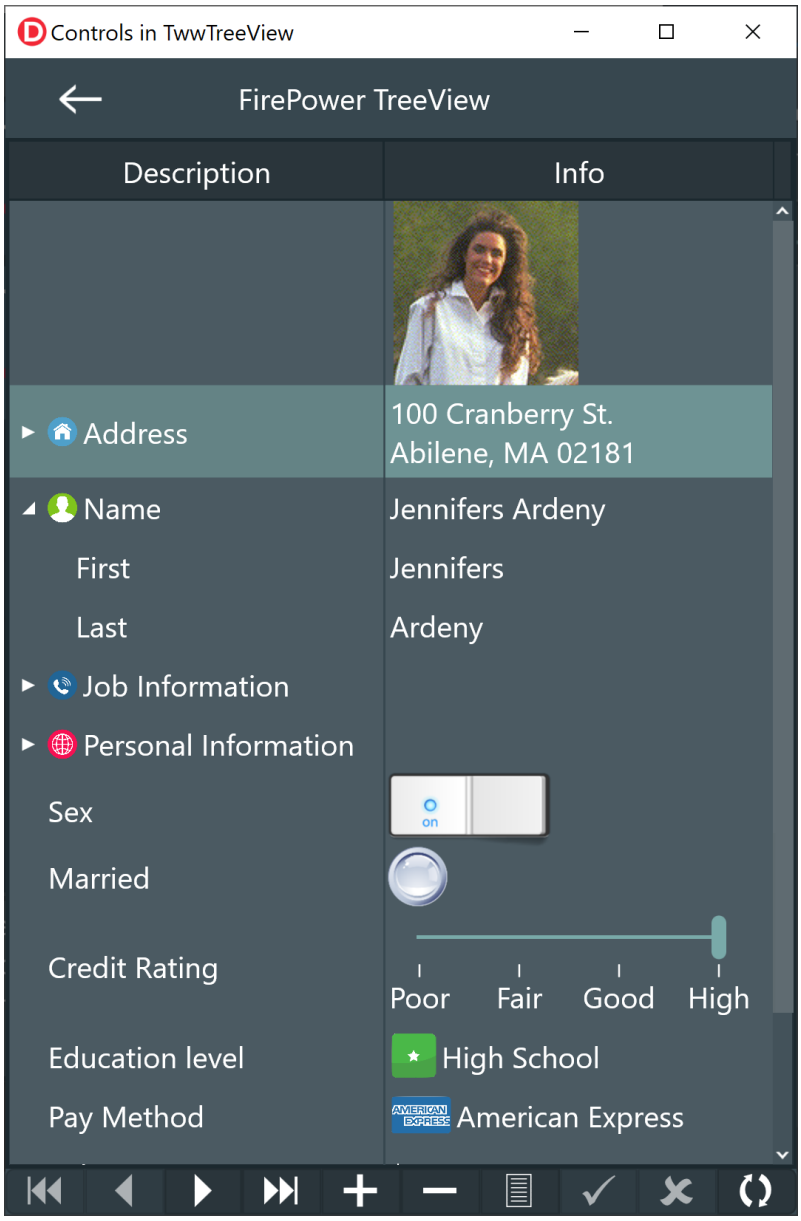
Use the flexible new TwwTreeView component to display or edit information in a hierarchical display. The FirePower Tree is built from the ground up to be fast. Load 10000 nodes in a fraction of a second, even on mobile devices. Your nodes can be variable height. Supports checkboxes, images, and automatically uses the 'TreeView' style in your stylebook. You can also use this control to edit hierarchical data (such as the IDE object inspector) with the ability to embed different FirePower controls into each row.

Setting up the TwwTreeView to edit data

To setup a TwwTreeView for editing data from a table, set the following properties

1. *ColumnEdit.Visible*. If you do not wish to show the editing column, set *ColumnEdit.Visible* to false.
2. Assign *DataSource* to the binding source that contains your data (such as a *TBindingSourceDB*). If you don't already have one on your form, you can right-click your dataset and select *Add BindSource*
3. Then to assign your treeitems to edit your data, dbl-click the component from the IDE to bring up the items editor where you can add your fields.
4. See the demo form *TreeViewWalkthru* that is part of our maindemo to see a complete example of using the tree to edit a record of your dataset.
5. Optional: Please note that this demo uses the *CustomFormat* property of the TwwTreeViewItem to automatically concatenate the strings of multiple fields to display in the tree when grouping related items together. The beauty of this is that you can see your data at design time, and do not need to create calculated fields in your dataset to compute these values. You should also set the ReadOnly to true so that focus does not go into this row/item. For an example, see the items in the tree captioned 'Name' and 'Address' form our demo mentioned above. Please note that you do not need to do this step if you are content with not seeing a summary of the child items data.

Configuring the type of control to edit with



To configure the control type for a tree item, use the *ControlType* and *ControlAttributes* properties of the *TwwTreeViewItem*

Auto-sizing of Caption and Edit columns

Use *AutoSizeColumns* to configure the tree to always fully utilize available space and adjust the editing column's width based on the width of the tree.

Assign images to the caption column

To assign images to the caption part of the tree, assign your images property in your *TwwTreeView* followed by assigning the ImageName property of your *TwwTreeViewItem*.

Enabling multi-selection checkboxes

To enable checkboxes in the caption column, set *ShowCheckboxes* to true. This will enable checkboxes for all columns. To individually configure checkboxes for each item, assign the *TwwTreeViewItem.CheckboxType* property. To check with your code if a item is checked, then use the *IsCaptionChecked* property of the *TwwTreeViewItem*.

Common Methods

To expand or collapse the entire tree use the *TwwTreeView's ExpandAll* and *CollapseAll* methods.

Adding Drag/Drop

If you wish to allow drag/drop between items on desktop platforms, then set *AllowDrag* to true

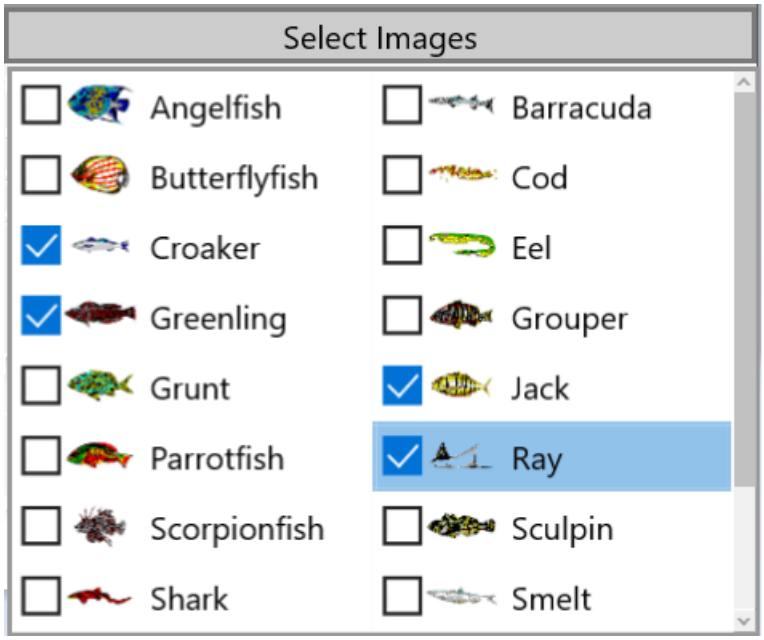
Adding tree items with code.

To add items with code use the following syntax. This will add 100 notes with 5 children in each. Make sure that you enclose your tree changes within BeginUpdate/EndUpdate or you may encounter unexpected painting issues.

```
var item, childitem: TwwTreeViewItem;
  i,j: integer;
begin
  wwTreeView1.BeginUpdate;
  for i:= 1 to 100 do
  begin
    item:= wwTreeView1.Items.Add;
    item.Text:= inttostr(i);
    item.EditText:= inttostr(i);
    for j:= 1 to 5 do begin
       childitem:= item.Items.Add;
       childitem.Text:= inttostr(j);
       childitem.EditText:= inttostr(j);
    end
  end;
  wwTreeView1.EndUpdate;
```

Combo Enhancements

New enhancements in this version support customization of the drop-down icons and support for multi-selections from the dropdown list using the TwwAdvComboEdit



Customizing the dropdown icon

Use the ButtonImages and ButtonImageName property to override the icon used for the dropdown button.

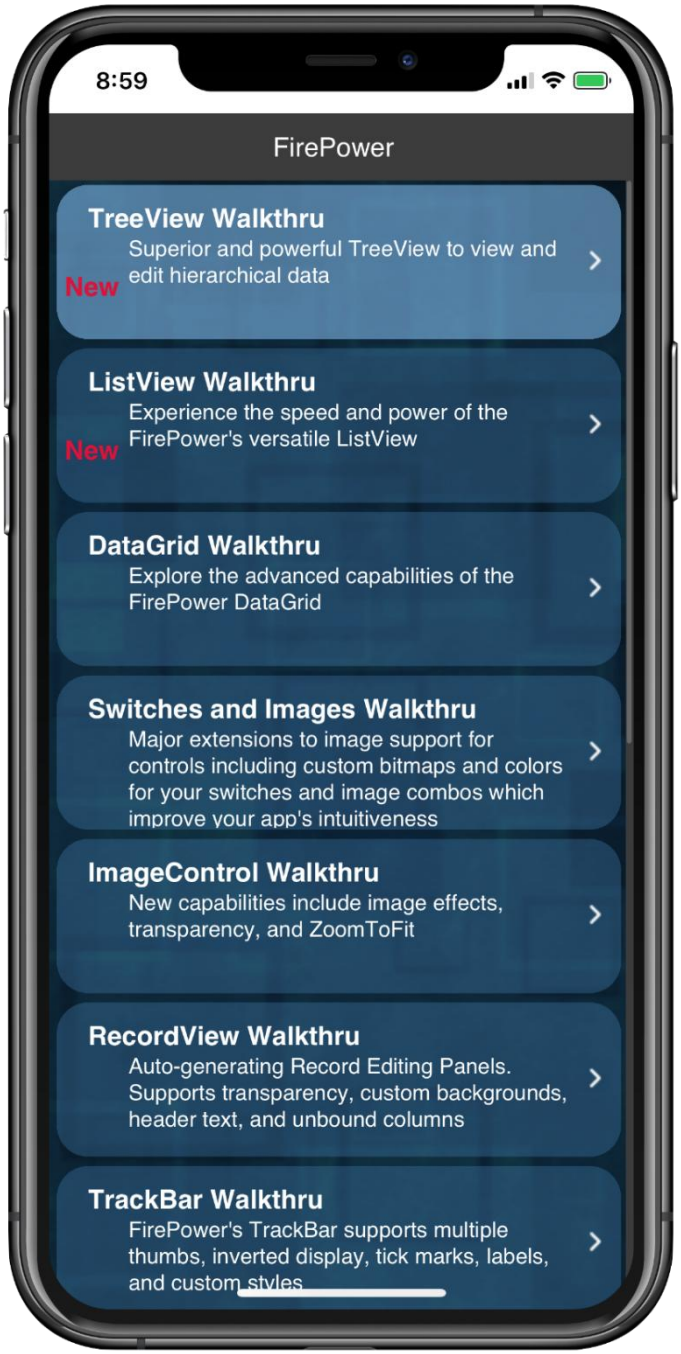
Enabling Multi-Selection Checkboxes

Set ShowCheckbox to true. This will enable checkboxes in the dropdown list. To access which items are selected, you can refer to the Item's Selected property. For instance this code shows each selection that is checked

```
for i:= 0 to wwadvcomboedit1.Items.Count-1 do
begin
  if wwadvcomboedit1.Items[i].selected then
    showmessage(wwadvcomboedit1.Items[i].StoredText);
end;
```

Select predefined listbox layouts for the dropdown list

Combos dropdown list can have multiple layouts using the new property ListboxLayoutName. You can also select this at design time by right-clicking the component and selecting *Select Item Layout*



TwwLayoutGrid modernization

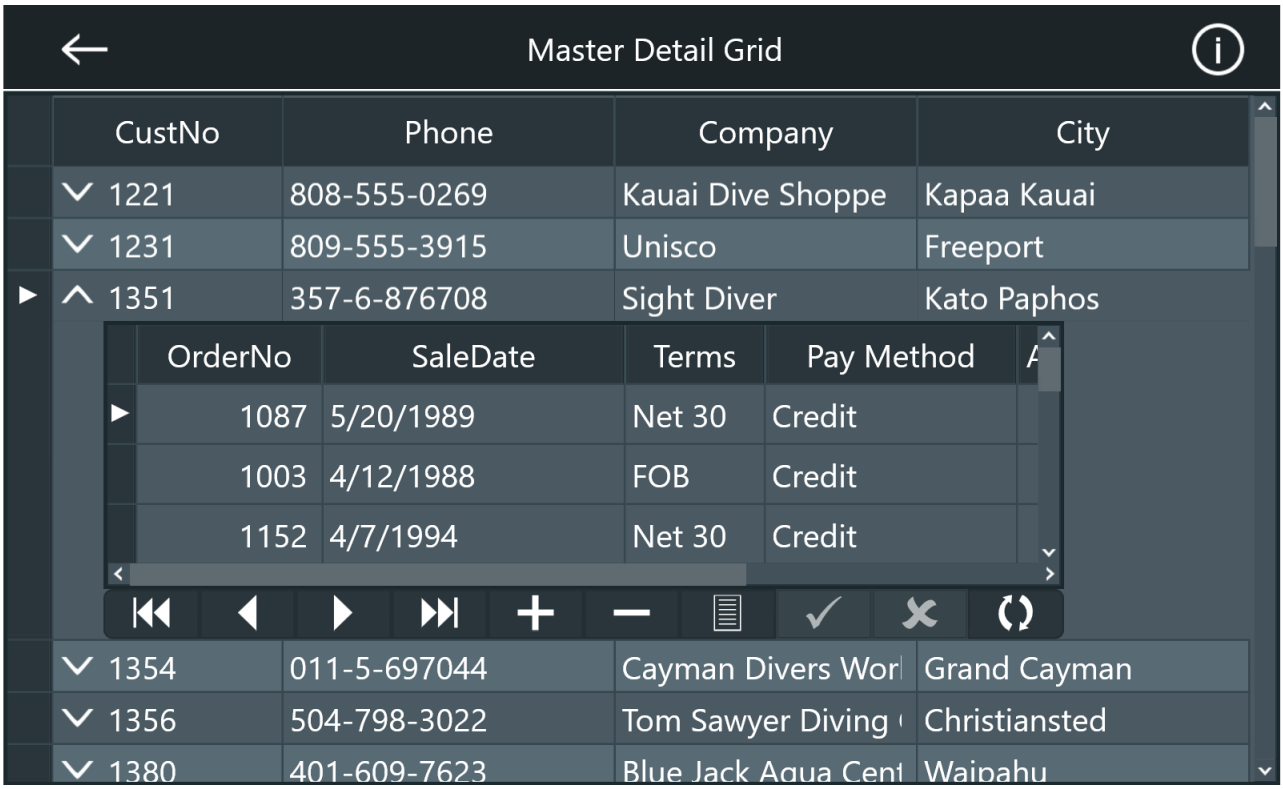
Support rounded rectangles and spacing for items using the new *OverrideStyleSettings.Frame* property. Assign *OverrideStyleSettings.Frame* to change the frame properties a record is presented by in the LayoutGrid. Normally this defaults to a simple rectangle, or just lines separating the records. By assigning these properties, you can change to a rounded rectangle and also change the colors. To enable custom framing, you minimally must set the *Frame.Enabled* to true and the Color property to a non-null value. You may also want to configure the frame's *ColorOpacityPercent* and the *SelectionColor* and *SelectionColorOpacityPercent* properties. Note that when framing is enabled, the lines are no longer drawn. You can review the *mainunit.pas* file in the *maindemo* application to see an example of framing enabled

TwwDataGrid now supports background images

Use the new ImageControl property to select a background image for the grid. You can use the ImageControl's properties to have ultimate flexibility on the background image including brightness, drawing style, stretching, tiling and zooming.

See Revision History for changes in FirePower

https://www.woll2woll.com/firepower-revisions



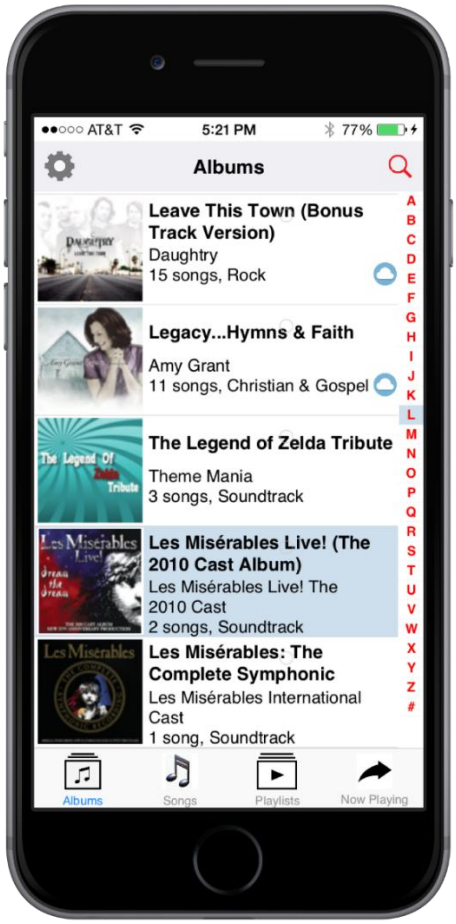
With FirePower X, embed grids within a grid to seamlessly display details of a record intuitively.

Demos

Please review the demos located in your public documents folder under. They are located in your public documents folder under

\Users\Public\Public Documents\FirePower\15.0\demos\23.0

We have included two complete demo projects (for both Delphi and C++ Builder) that include all the demos, ***MainDemo*** and ***MainDemo_sqlite***. They are identical except MainDemo_sqlite uses a sqlite database for the data where MainDemo uses in-memory data stored in the form's fmx files. See Project | deployment to see the specifications of the files for deployment of the sqlite tables to iOS and Android.



Demos

These screenshots are taken from actual demos included with FirePower.

Our exciting FirePower suite of components allow you to develop applications for Win32, Win64, Mac (OSX64), Android, and iOS devices (32 and 64 bit) and simulators using Embarcadero's RAD Studio. Similar in capabilities to our award winning InfoPower VCL library, our FMX components integrate seamlessly with your existing data and are truly data-aware components.

Rapidly build powerful business applications

The FirePower component suite includes the critical and necessary user interface components for building professional desktop and mobile applications using RAD Studio. In particular those who are building business applications will directly benefit as the controls display your data quickly and in a natural way. FirePower is built from the ground up so that your efficiency, look and feel, and power are not compromised when moving your applications to the mobile space.

No compromise when moving to the mobile space

Central to our component suite design are two powerful and flexible grid controls. Our FirePower grid components are designed to be incredibly fast, efficient, powerful, and flexible. They are designed to perform well on desktops, but take great advantage of the mobile interface, giving the end-user a natural experience in style, performance, and ease of use.

Suite of versatile user-interface components

Also in FirePower is the TwwRecordViewPanel component giving you a flexible layout component specifically to edit a record. The component dynamically creates an editable panel based on your dataset's field properties. It removes the tedious job of building custom record editing forms, and lets you focus on which fields you want edited and in what order.

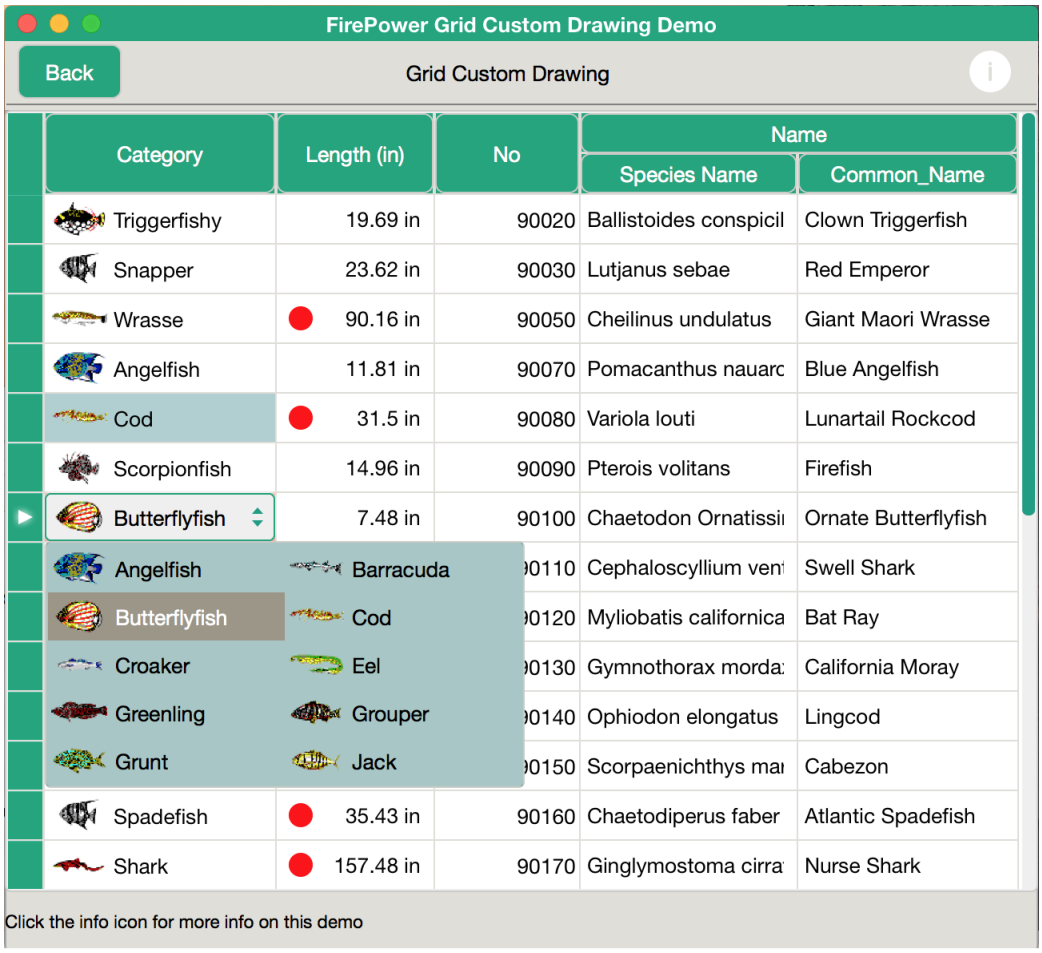
There are also an array of edit and input controls included with FirePower that will significantly enhance your application, as well as components to assist your end-users in filtering and searching for data.

Prebuilt applications that use FirePower that you can run right now on your own devices:

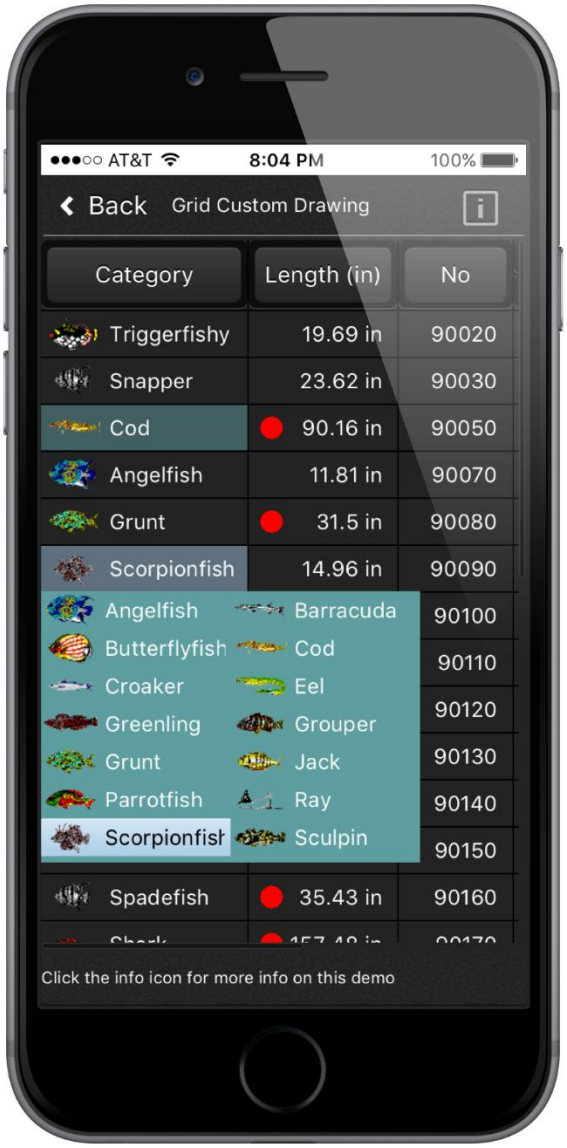
To download applications already built with FirePower, visit the App store for your device. You'll find links at bottom of the page www.woll2woll.com/firepower

Demo for FirePower in iOS store at

https://itunes.apple.com/us/app/firepower-delphi/id1102273918?mt=8



Included FirePower demo running on OSX



FirePower's Masterpiece Grid

Central to our component suite design for FireMonkey is our greatly enhanced data-aware grid which is complemented with a library of other components. Our FirePower grid component is mobile optimized as it is designed to be incredibly responsive and natural for the end-user, and powerful and flexible for the developer. Here are some of the capabilities of this grid.

- **Expandable Grids -** Show details of a record within the grid by embedding the TwwExpander as a custom control of a column.

- **Embed custom controls** such as buttons, combos, imagecombos, checkboxes, switches, lookupcombos, datepickers, trackbars, progressbars, and images into the grid. Absolutely no coding required for this. Also you can dynamically determine when custom controls are displayed based on the value of data for that record.

- **Performance**: Data is buffered so that the grid only loads records that it needs to display. This allows for fast grid display that is not crippled by the number of records or columns in the dataset. In fact, our FMX grid rivals in performance to our super-fast VCL grid.

- **Smooth scrolling** – The mobile space relies upon touch, acceleration, and momentum for scrolling. The DataGrid implements smooth scrolling using all of these factors giving your mobile apps a natural responsive experience.

- **Clickable column headers** with built-in support for sorting (ascending or descending). The buttons for the column headers are automatically painted in the native style.

- **MultiSelect support:** Allows you to select multiple records with a checkbox or a switch in the grid and operate on them as a whole

- **In-memory support -** Use with in-memory datasets such as the TClientDataSet to have a grid display data independent of any physical file.

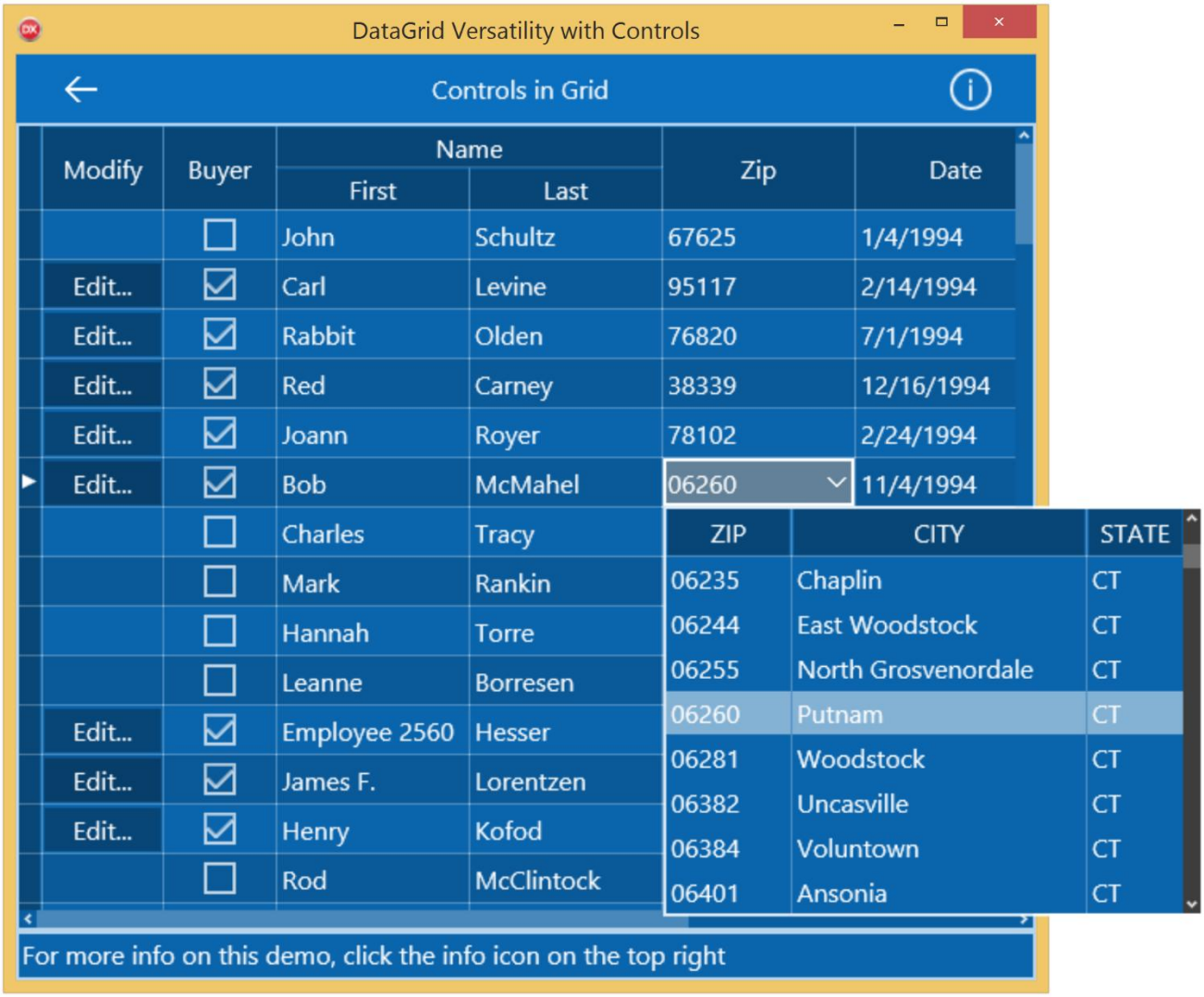
- **Fixed column support**, including editable fixed columns.

- **Export Data** : Methods to export or email your grid's data to various formats for use with other applications for both the desktop and mobile platforms. Can be used in conjunction with filtering so that only certain records are exported.

- **Codeless Validation constraints**: Supports InfoPower's robust validation languages for enforcing data constraints. The developer can use either picture validation masks or regular expression validation masks. See section 3 below for more information on the validation languages.

- **Flexible design time columns editor:** Our columns designer is significantly enhanced over our VCL counterpart and you'll be able to accomplish even more tasks without any coding.

- **Integrate styles seamlessly** with no effort on your part. Just change the style and the grid recognizes the changes. You can still override the grid's fonts and colors when needed. You can even change your whole app look to a windows 10 style without any coding. In addition when using touch screen Windows 10 devices, the grids and embedded controls are also touch enabled.



Included FirePower demo running with Win10 style selected

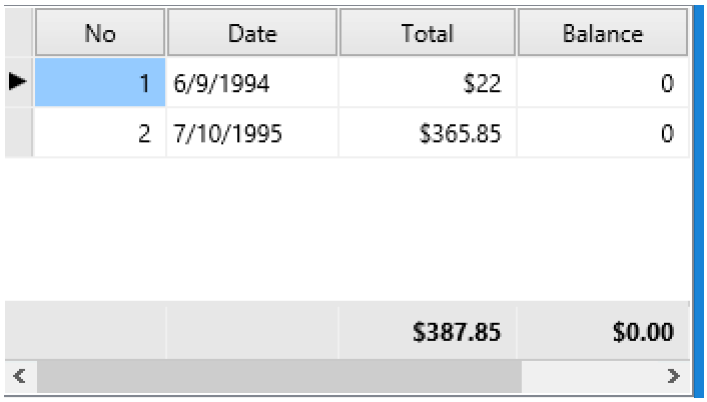
- **Flexible painting :** Customize colors/fonts based on a record or cell basis. Can display custom controls on a record by record basis instead of for every record. Numerous painting options including alternate row colors, highlighting of entire rows. FirePower also has many events for custom drawing of the cells so that your customizations are limitless.

- **Natural keyboard handling –** Convert carriage returns to tabs, virtual keyboard supports automatic vertical shifting of edit control for display,

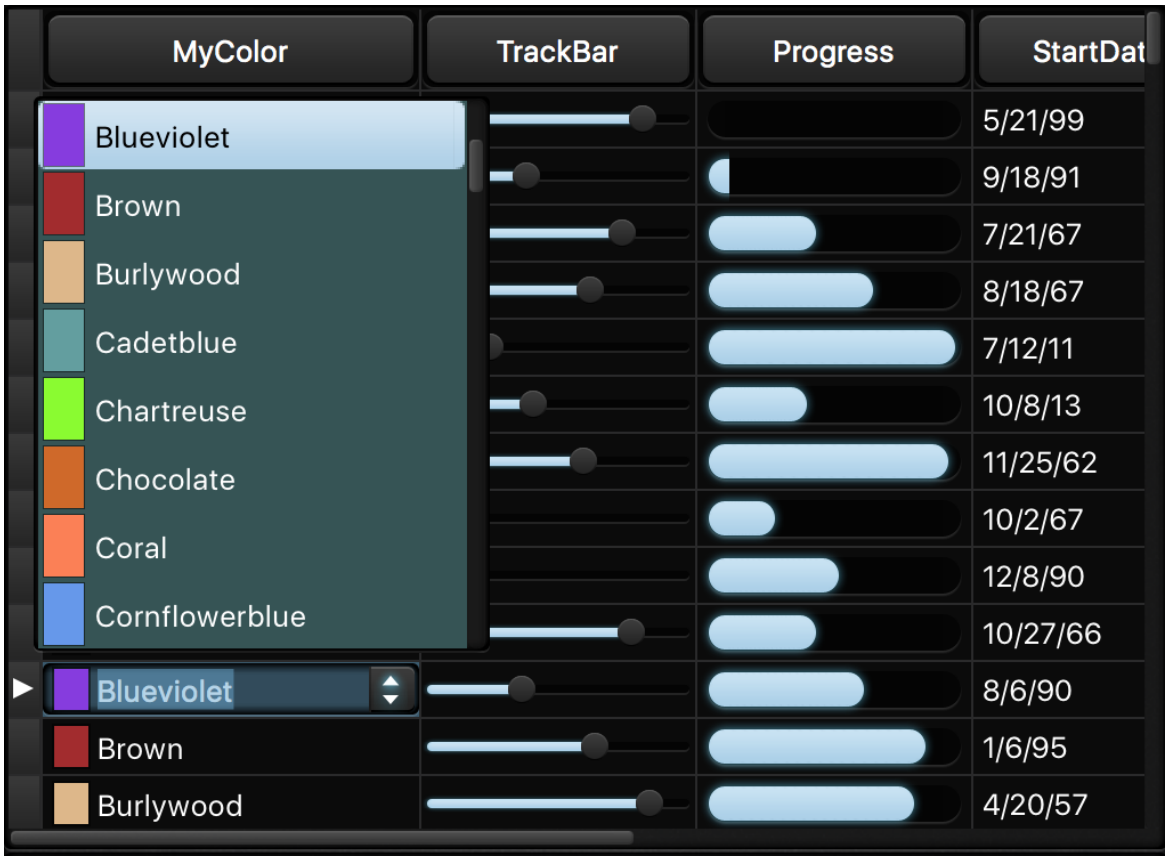
- **Filtering** - Use the grid with our new ***TwwSearchBox*** to get seamless filtering. You can automatically allow the text to be searched on one or multiple fields and have the grid highlight the cells that match the text.

- **Grids can be used FireDAC, UniDAC, in-memory data structures or 3rd party database engines:** Demos included to show you how you can display, edit, and sort data with in-memory data structures and have the data independent of any physical file.

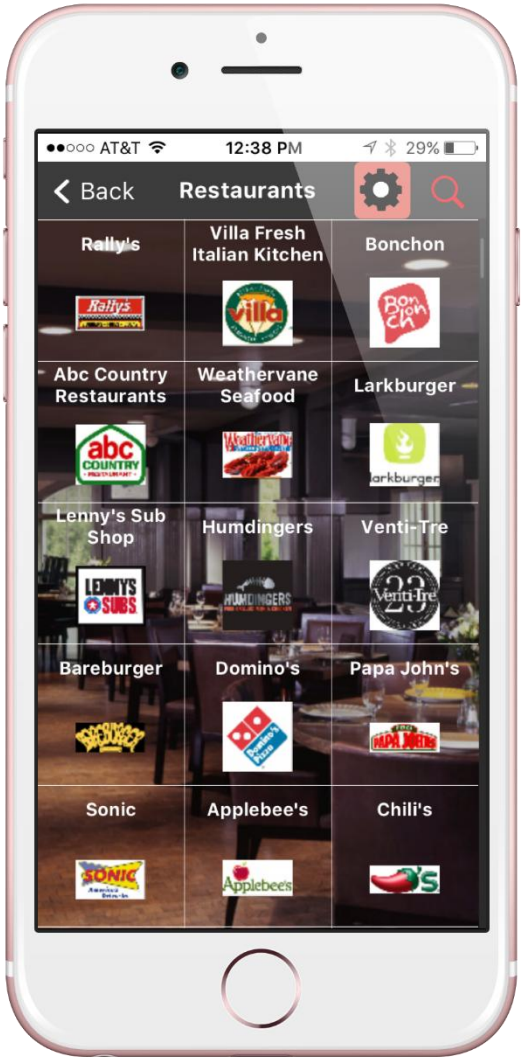
- **Footer Support –** (See the MainDemo)



- **ColorCombo/Trackbar/Progress Bar Support –** (See the MainDemo)



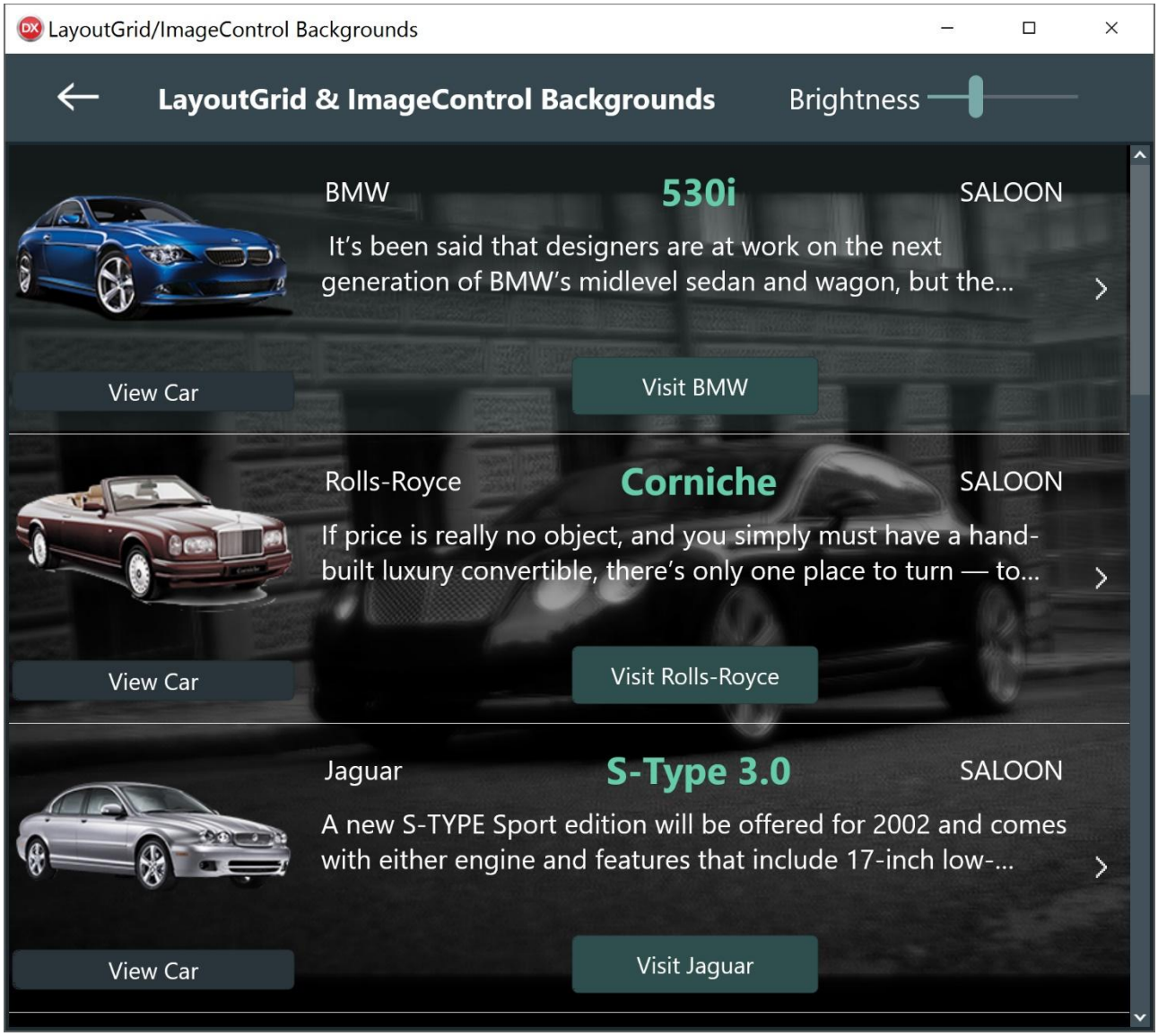
Versatile LayoutGrid/ListView



The versatile and powerful TwwLayoutGrid is the ideal ListView control, which allows you to visually design the layout of your data by dropping in embedded controls into a panel that at runtime are repeated vertically, horizontally, or multi-dimensionally. This is particularly useful for the mobile space where a more open layout allows the developer to express the user-interface more intuitively. It also allows you to represent your data naturally so that your limited mobile screen real-estate is not consumed. The standard vertical grid may not always be the

ideal interface for multi-record display. Also includes touch support, automatic editing, and numerous smart events that will allow you limitless expressions of your application.

Blazingly Fast!! - You will be impressed by the outstanding performance of this component, whether on the desktop or mobile space. Your end-users will love the natural usability and intuitiveness of your applications when using this component.



RAD Development with WYSIWYG in the IDE

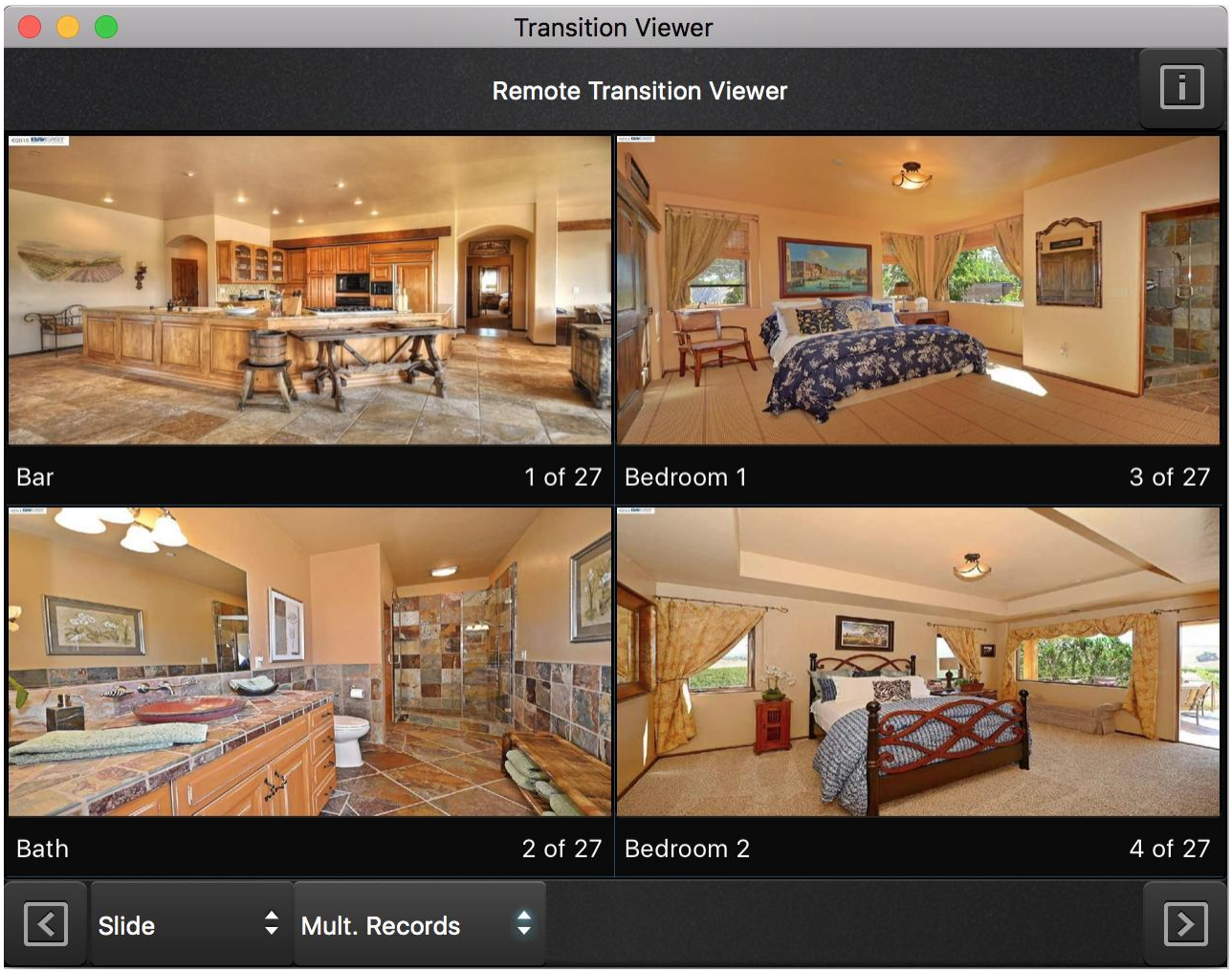
TwwLayoutGrid design enhancements to support complete WYSIWYG during design time. You can drag/drop unlimited amount of controls and use them recursively for the optimal layouts, and you'll see exactly how it looks

in the IDE with your data without having to run your application. You'll also be able to add stunning elegance to your apps with transparency effects and automatic background tiling. Fully integrated with styles.

Use with or without a physical database

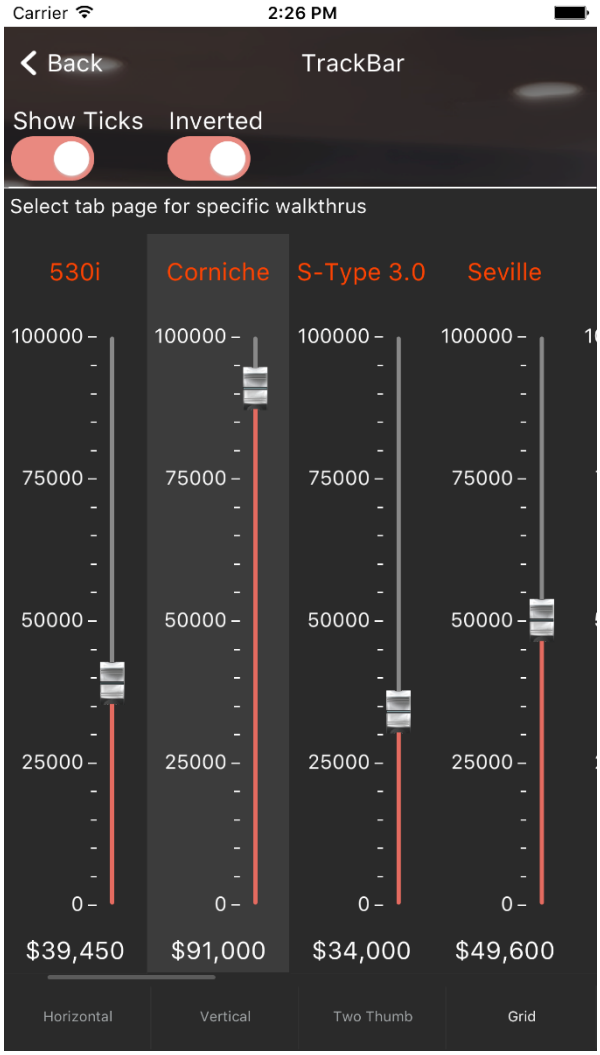
The layout grid has the flexibility to attach to your own memory structures or you can bind it directly to a table. You can also just define a TPrototypeBindSource's data structure, and then dynamically fill the display dynamically with code.

Transition Viewer



Use the TwwLayoutGrid as a Transition viewer for one or more records or images. Supports 22 transition effects including BandedSwirl, Blind, Blood, Blur, Bright, Circle, Crumple, Dissolve, Drop, Fade, Line, Magnify, Pixelate, Ripple, RotateCrumple, Saturate, Slide, Shape, Swirl, Water, Wave, Wiggle. For a demonstration see [Transition Viewer Demo]

Flexible and Sophisticated TrackBars



FirePower's professional TrackBar control has the most flexible and sophisticated capabilities for FireMonkey. You can select ranges, provide custom thumbs, invert the trackbar, display ticks and labels, and much more.

Supports multiple thumbs

Display two thumbs so that the user can select a range instead of just a value and custom images for your thumbs.

Custom Thumb bitmaps

Choose your own thumb to customize the elegance of your applicatoin

Inverted display

Reverse the display of the TrackBar so that the maximum value is on the top or the left instead of the bottom or the right.

Embeddable into FirePower Grids

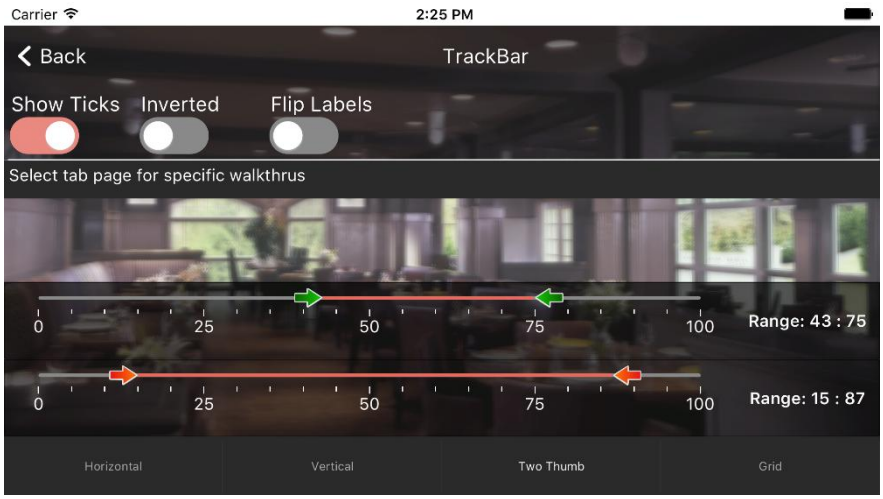
The TrackBar can also be embedded into FirePower grid controls as in the following screen.

Custom Styles

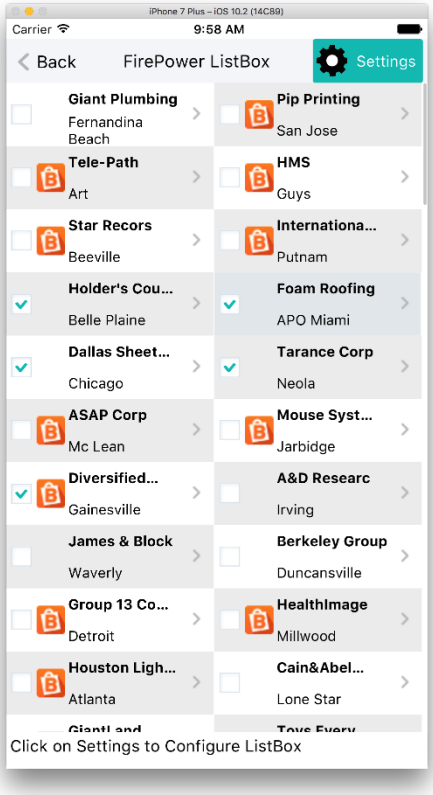
Use with any custom stylebook and the FirePower trackbar will automatically use the style colors and display

Display of Tick marks and labels

- Supports the display of helpful tick marks and numeric labels to assist them when they are manipulating a trackbar.



Lightning quick and powerful ListBoxes



FirePower's TwwListBox gives you powerful, fast, and efficient listboxes. Even on mobile devices, you will be able to display and scroll almost instantaneously thousands of items.

Powerful and flexible Designers for Codeless Development

Development is extremely intuitive and natural with this component. Flexible Design tools allow you to visually design your ListBox layout with controls such as *multi-select checkboxes*, *imagelists*, *accessory icons*, and *detail* text. Each embedded control can also be customized individually for maximum flexibility. The display of multiple columns in your listbox is also seamless and efficient.

Stock Layouts

The Listbox has predefined layouts so you can rapidly configure their look and feel.

Incredibly fast loading and response times

Load thousands of records at the blink of any eye, and still able to scroll smoothly even on mobile devices.

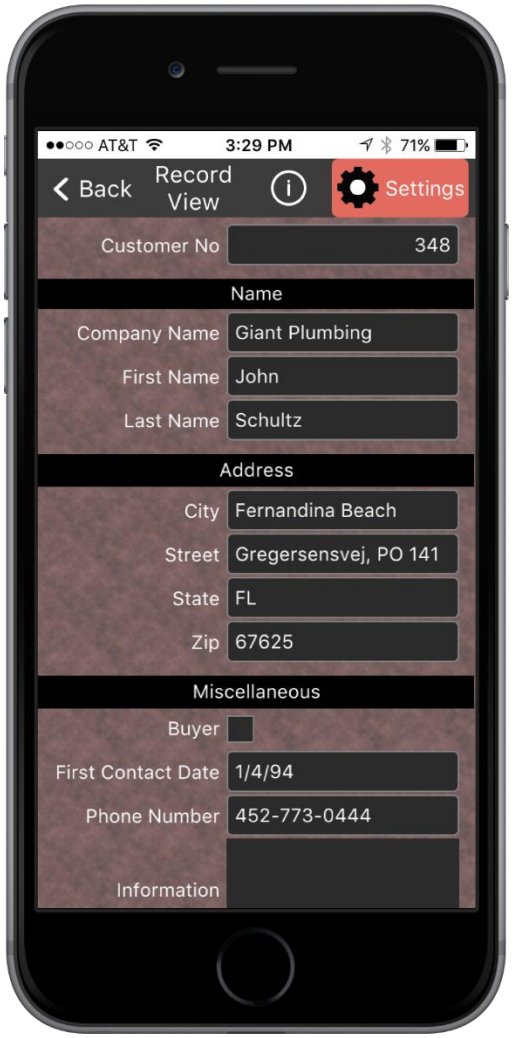
Directly bind to data sources or your in-memory structures

Add listbox items through design editor, code, data sources, or your own in-memory lists.

Dynamically created data-entry forms

FirePower includes the flexible TwwRecordViewPanel component giving you a flexible layout component specifically to edit multiple fields in a record. The component dynamically creates an editable panel based on your dataset's field properties. It removes the tedious job of building custom record editing forms, and lets you focus on which fields you want edited. This is the ideal companion for the grids so that you can quickly create and customize your user's record editing abilities. Our new version also supports associating fields with advanced combos, colorcombos, trackbars, progressbars, checkboxes, and switches.

The RecordView is a great time-saver for FireMonkey forms as it adapts to the size of your target device based on your property settings. You can specify a vertical or a horizontal layout of the fields to edit. You can also at design time just drag and drop the fields in the order you want. The component can even dynamically add all the fields in your dataset if you desire, removing the need to manually create data entry forms.



Virtual Keyboard Auto Scrolling - Version 10 adds automatic scrolling of the edit controls to keep them in view when the virtual keyboard pops up. It also supports more custom controls such as the TrackBar, ProgressBar, and ColorCombo.

DataSource not required

Does not require a datasource so you can use it to edit user configuration settings or program options data.

Grouping of related fields

Create separate subgroups within the record view

Superb Validation language

FirePower provides two independent but expressive languages to validate your user's data entry and protect the integrity of your database.

As-you-type validation is also supported, so the end-user can be visually notified of the validity of their input using user-definable error colors.

Regular Expression edit masks

A regular expression is a special text string for describing a search pattern. Regular expressions are one of the most powerful ways to define a set of rules, and are widely used across many different programming tools.

FirePower checks the user's input against regular expressions masks, and allows you to prevent the user from entering invalid data.

Picture edit masks

FirePower gives you incredibly flexible and expressive picture masks which greatly assist in automatically validating your user's input. FirePower's masks duplicate the Picture function that's been available in Borland's Paradox relational database product, providing the power of a full mask language instead of just a mask template. Picture masks greatly assists the end-user during data entry by intelligent auto-filling of characters where appropriate as well as informing them if they have incorrectly entered any data. This allows data entry to be faster and easier. Here are some examples of the power of picture masks.

Street Address Capitalization

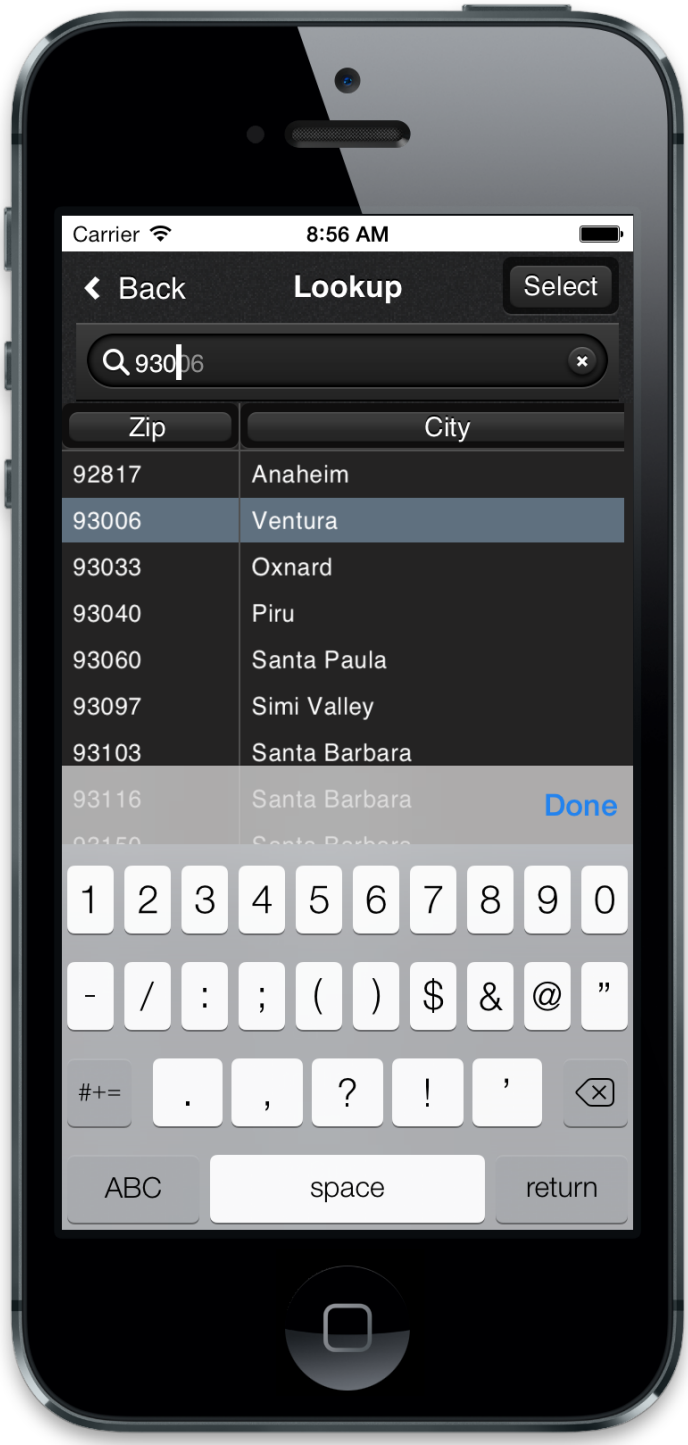
The first letter of each word is automatically capitalized. For instance, if the user enters “235 quinault way”, FirePower conveniently converts the input to “235 Quinault Way” as they are typing.

Masks which contain optional sequence of characters

Such as an optional zip code suffix in a zip code. If the user does not match the format defined by your mask, you can inform the user visually or prevent the control from losing focus.

Intelligent auto-filling of characters as the user types

The user's keystrokes can be minimized as the mask language can fill in the static characters. For instance with a SSN mask of ###-##-####, the dashes (-) can be automatically filled in so that the user just needs to type in the numbers.



Advanced LookupCombo and LookupDialog Controls

FirePower gives you the most flexible component for selecting entries from a lookup datasource. The components are not compromised by smaller form factors in the mobile space, and allow users to quickly lookup and locate the data they want. Here is some of what this powerful component can do.

Quicken style incremental searching

Supports the 'Quicken' style display of the matching value, by simultaneously searching and displaying the matching text in the search control.

Multiple Columns in the DropDown Box

Select any number of fields to be displayed in the drop-down list along with defining their display width and optional titles.

Sorting flexibility

The values in the drop-down list are sorted in the order of the first field you select to be displayed, if it's a secondary index field, instead of being sorted in primary key order.

Embed into FirePower's DataGrid and LayoutGrid components:

The component can be used in a TwwDataGrid component to replace any multiple-choice type of field in the grid, giving your end-users sophisticated lookup and fill capabilities within the grid.

Use unbound or bound

The component does not have to be bound, or assigned, to a table's field which gives you greater flexibility in using this LookupCombo for general tasks where a source table is not involved.

- **Auto-detection of lookupfields** in dataset for ease of setup

Date Control - TwwCalendarEdit

FirePower provides the most flexible and usable CalendarEdit control for Delphi. DateTimePickers support drop-down calendars or time selectors to assist the user in selecting a date.

Embed within FirePower's **DataGrid and LayoutGrid** components.

- Use with or without a database.
- Smart data entry : Auto-advances when enough characters have been entered, and auto-fills the date
- Display the date in the format of your choice using a display format mask. Also supports International date-time formats.
- Integrated with the platform's native date picker on the mobile space

Advanced Combo Controls (TwwComboEdit, TwwAdvComboEdit)

- Support ImageList so that images can be displayed in the combos alongside the text
- Mapped List support so that you can display a descriptive value and store your coded value
- Incremental Searching support as you type
- Supports multiple columns for the drop down display
- Embed into FirePower's DataGrid and LayoutGrid for graphical icons next to your text and drop-down list items

- Support for *ImageLists*, *ImageName*, multiple columns, and the ability to hide the text and just show the images in both the control and its dropdown list (See screenshot below). Hiding the text is useful if the graphical representation is sufficient information for the user.

TwwColorComboEdit

- Displays drop-down list of colors with color box
- Incremental Searching support as you type
- Can use live binding to an integer value in a database, or you can bind it to the color property of a control.

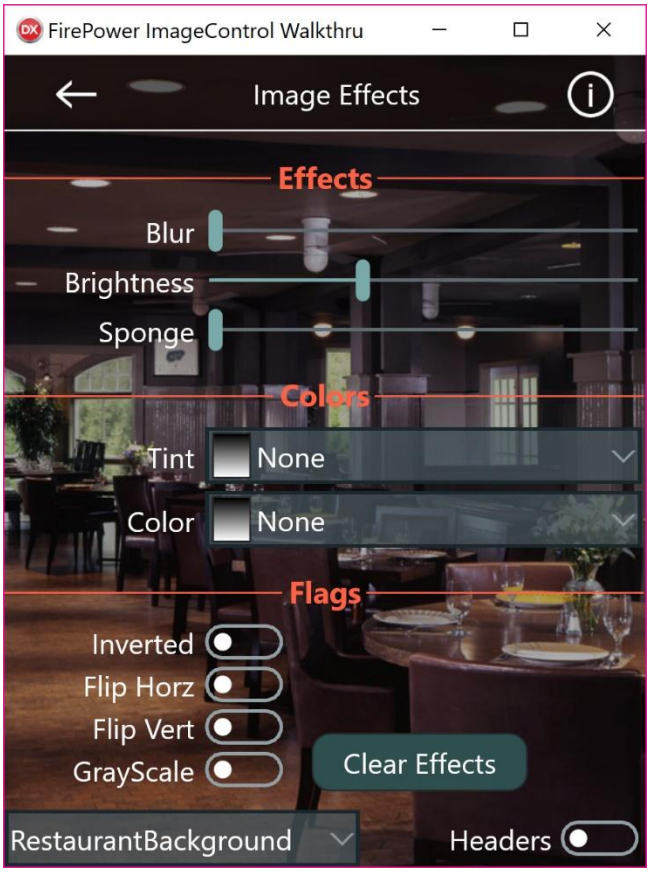
TwwButton

Supports custom coloring and lines and is embeddable inside FirePower controls and grids. Works consistently across all platforms.

Advanced ImageControl with custom effects

Use the ImageControl to add image effects such as brightness and tinting, and control how the overall image is painted. This control can also be easily used a background for FirePower's LayoutGrid and RecordView providing an elegant way to give your applications a 1stClass appeal. See the MainDemo for an example.

Support for images defined and loaded from an ImageList instead of being assigned. This is more efficient in memory and loading if you have the same being used in multiple places. You can also refer to an ImageName from an *ImageList* instead of just an *ImageIndex*

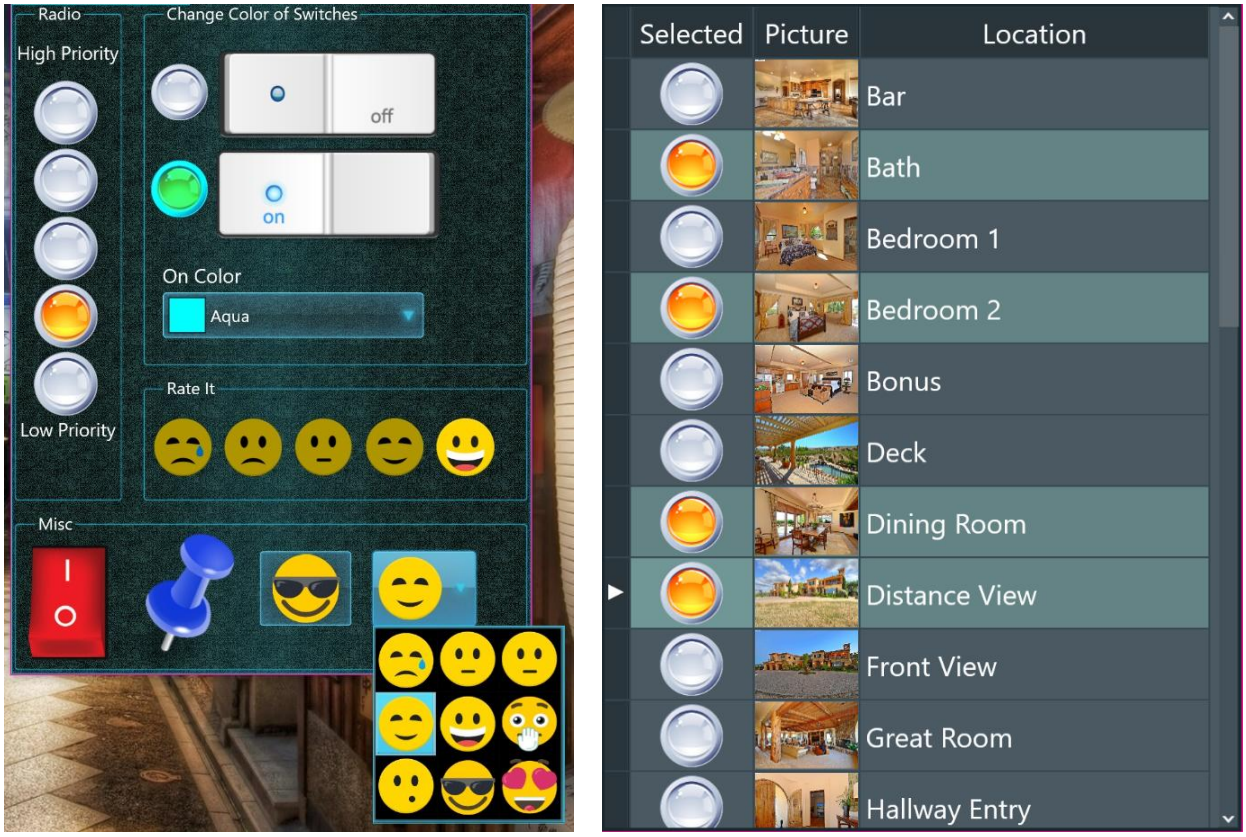


Progress Activity Dialogs (Mobile)

Mobile - Display activity wait dialog indicators for visual feedback to the user during lengthy operations. They run in the background so your painting activity is not stalled.

Enhanced Sharing Services for Mobile

Share one or more files through iOS/Android sharing service. This allows you to share via email, google drive, etc. You can also share other file types besides images. Also includes common methods for emailing and opening url links for both desktop and mobile devices.



New graphical customizable switches, graphic and button style combos, and graphical switches in grid. No code required!

Checkboxes, Switches

Enhanced versions of TwwCheckBox and fully customizable graphics for images displayed for TwwSwitch.

- The switch controls support custom images so that you can use it to display more intuitive controls for your end-user. Image can come from *ImageLists* specified with either a reference to either

ImageIndex or *ImageName*. Using *ImageName* is especially convenient so that your images are not dependent upon the order, but a more static reference such as the name of the image. Also support for custom effects and colors for each state of the image, such as checked, unchecked, or hot.

- You can assign a *GroupName* to a collection of switches to create graphical and custom radio butto.

Visual Filtering

FirePower has various controls for searching and filtering through your database. The TwwSearchBox can be used on the desktop or on mobile devices and allows you to define one or more fields that are searched through a single edit control. This is particularly useful on the mobile space where desktop space is at a premium. The TwwFilterDialog (currently desktop platforms only) is one of the most useful end-user components as it enables them with the ability to visually filter a dataset, modify the where clause of an existing SQL query.

Simple for the end-user to use:
Even though the dialog is capable of sophisticated SQL generation, the dialog is simple to use as it completely hides the filtering and SQL details from the end-user.

Unmatched filtering power:

Your end-users can specify a search value, or a range, for any number of fields contained within the dataset.

WildCard Filtering within Fields

Select a specific type of data match to be performed within the field, such as "From beginning of field", "Anywhere within the field", and "Exact match".

Special customizable keywords

Specify keywords such as "AND", "OR", "NULL" to specify multiple filter criteria for each field, such as... John OR Paul. FirePower also allows the end-user to easily see all non-matching records.

Filter memo fields

See http://www.woll2woll.com/Ordering.html for further details on ordering FirePower, and special upgrade pricing.

Woll2Woll will also issue more rapid updates to keep in sync with the changes in Rad Studio. Our maintenance contract will allow you to receive all our major updates for up to one year.

Known issues:

When using the new multi-device view for any form, and selecting any device for the very first time, first save your form before editing with the FirePower design tools. There is a known issue with the Delphi IDE that can cause

access violations if you do not save your form first after selecting the new device. Once you have saved your form, you do not need to repeat this process. If you are using a device that you have edited at any time in the past for this same form, then this is not necessary.